基于逐维反向学习的动态适应布谷鸟算法 *

黄闽茗 [1a, b]，何 庆 [1a, b+]，文 熙 [2]

(1. 贵州大学 a. 大数据与信息工程学院; b. 贵州省公共大数据重点实验室，贵阳 550025; 2. 河北工业大学 人工智能与数据科学学院，天津 30000)

摘 要：为了解决布谷鸟搜索算法(CS)寻优精度不高、收敛速度慢、后期搜索活力不足以及处理高维优化问题时存在维间干扰等缺陷，提出了逐维反向学习策略的动态适应布谷鸟算法（DA-DOCS）。首先，对选择更新后的解进行逐维反向学习，减少维间干扰，扩大种群多样性；然后，使用精英保留方式评价该结果，提高算法寻优能力；最后，充分利用当前解的信息进行动态适应的缩放因子控制，引导解快速收敛，提升算法搜索活力。实验结果表明，该算法相比较于标准布谷鸟搜索算法，寻优精度、收敛速度以及后期搜索活力有所提高，与其他改进算法相比也具有一定的竞争优势。

关键词：布谷鸟搜索算法；反向学习；函数优化；维间干扰；动态适应

中图分类号：TP301 doi: 10.19734/j.issn.1001-3695.2018.10.0725

Dynamically adaptive Cuckoo search algorithm based on dimension by opposition-based learning

Huang Minming[1], He Qing[1+], Wen Xi[2]

(1 *a. College of Big Data & Information Engineering, b. Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang* 550025, *China*; 2 *College of AI & Data Science, Hebei University of Technology, Tianjin* 300000, *China*)

Abstract: However, there are still some shortcoming in cuckoo search algorithm(CS) , such as low convergence precision, slow convergence speed, Weak search vitality and interference phenomena among dimensions when dealing with high-dimensional optimization problems. Dynamically Adaptive Cuckoo Search Algorithm Based on Dimension by Opposition-based Learning(DA-DOCS) was proposed, Firstly, the selected solution updated for dimension-by-dimension by Opposition-based Learning, this result reduced interdimensional interference and expanded population diversity. Then the method of elite retention was used to evaluate the results and improve the search ability of the algorithm. Finally, the information of the current solution was fully utilized to dynamically adaptive the scaling factor control to guide the solution to converge quickly and enhance the search vitality of the algorithm. The experimental results show that compared with the standard cuckoo search algorithm, the proposed algorithm has improved convergence precision, convergence speed and search vitality. Compared with other improved algorithms, it has certain competitive advantage.

keyword: Cuckoo search algorithm; opposition-based learning; function optimization; interdimensional interference; dynamically adaptive

0 引言

布谷鸟搜索算法(Cuckoo search algorithm, CS)[1]是由 Yang 和 Deb 于 2009 年提出的一种新的全局搜索算法，该算法源于对布谷鸟育雏行为以及鸟类的莱维飞行（Lévy flights）行为的模拟。由于算法具有高效、参数少、易于实现等优点，已经在工程优化和函数优化问题中得到了广泛的应用，成为启发式智能算法领域的一个新的亮点[2,3]。但 CS 算法也存在收敛能力差、搜索活力不足[4,5]等缺点，因此，国内外学者陆续对算法做了进一步的研究和改进。

王凡等人[6]通过分析 CS 算法的 Markov 链模型，研究算法的状态转移过程，证明了 CS 算法的收敛性。陈华等人[7]结合 Logistic 模型改进 CS 算法，自动调节步长控制因子和发现概率的大小，提高了全局搜索能力。实验表明，改进算法具有较好的性能。Pauline 等人[8]提出一种双亲交叉的自适应布谷鸟算法，通过实验验证了该算法的有效性。马卫等人 [9]通过构建精英种群候选解池与 Powell 局部搜索策略相结合，通过实验证明改进算法具有一定的竞争优势。

上述研究成果丰富了 CS 算法的相关改进工作，都较好地改善了 CS 算法的收敛速度和收敛精度，但大多数改进算法都采用整体更新再评价策略，即先更新新解的所有维度信息，再根据目标函数进行评价，这样的评价更新方式使得维间相互干扰，将恶化解的收敛效率和收敛速度。但是王李进等人[10]提出基于逐维改进 CS 算法，采用逐维更新评价策略，强化进化维的信息，减少维间干扰；实验结果验证了该算法的有效性，但在处理高维复杂函数时，逐维策略在收敛前期将耗费大量的评价次数，导致收敛效率变慢，收敛后期活力不足。

收稿日期：2018-10-25；**修回日期：**2018-12-03 **基金项目：**贵州省科技计划项目重大专项项目（黔科合重大专项字［2018］3002，黔科合重大专项字［2016］3022）；贵州省公共大数据重点实验室开放课题（2017BDKFJJ004）；贵州省教育厅青年科技人才成长项目（黔科合 KY 字［2016］124）

作者简介：黄闽茗（1995-），女，硕士研究生，主要研究方向为进化计算，数据挖掘；何庆（1982-），男（通信作者），副教授，博士，主要研究方向为大数据应用、进化计算（qhe@gzu.edu.cn）；文熙（1995-），男，硕士研究生，主要研究方向为进化计算，智能控制.

受逐维改进策略以及各混合改进算法[11~13]的启发，本文主要针对标准 CS 算法收敛速度偏慢、求解精度较低、搜索活力差以及高维复杂函数难收敛等问题，提出了一种基于逐维反向学习的动态适应布谷鸟搜索算法（dynamically adaptive cuckoo search algorithm based on dimension by opposition-based learning, DA-DOCS）。首先在算法迭代过程中利用逐维反向学习策略，以便减少各维间的相互干扰，通过反向学习扩大搜索空间，能够提高搜索效率；然后利用精英保留方式评价逐维反向学习后的解，能够提高算法的搜索速度；最后利用当前解的信息动态适应控制缩放因子，动态引导解的收敛，提升寻优能力。本文选取八个标准测试函数进行实验仿真，结果表明，与 CS 算法相比，本文改进的 DA-DOCS 算法具有更好的收敛精度、收敛速度以及收敛效率，在高维度优化问题上，性能优于 CS 算法。与其他改进 CS 算法相比，DA-DOCS 算法搜索活力更强，寻优能力更优。

1　相关工作

1.1　布谷鸟搜索（CS）算法

布谷鸟算法其基本原理就是把蛋所寄生的巢穴位置映射为算法种群空间中的解，以寄生巢穴位置的优劣作为算法的适应度值。为了模拟布谷鸟的繁衍机制，算法设定了三个理想规则[1]：

规则 1 每只布谷鸟只产一个蛋，并随机放入所选择的寄主鸟类的巢穴中；

规则 2 每次进化都保留最优蛋的巢穴到下一代；

规则 3 可用的寄主巢穴数量固定，且寄主鸟类以概率 $R\in(0,1)$ 发现布谷鸟放的蛋。

根据规则，CS 算法初始化之后，通过莱维飞行更新巢穴的位置，公式如下[1]：

$$x_{t+1,i}=x_{t,i}+\alpha_0\otimes Levy(\beta) \tag{1}$$

其中：$x_{t+1,i}$ 表示巢穴位置更新后的位置；$x_{t,i}$ 表示当前巢穴位置；α_0 表示步长缩放因子，通常为 0.01[2]。

莱维飞行随机搜索路径公式如下：

$$Levy(\beta)\sim\frac{\Phi\times u}{|v|^{1/\beta}}(s,\lambda) \tag{2}$$

其中：u 和 v 表示标准正态随机变量；β 表示莱维飞行的控制因子，通常为 1.5[2]；Φ 的计算公式如下：

$$\Phi=\left[\frac{\Gamma(1+\beta)\times\sin\left(\pi\times\beta/2\right)}{\Gamma\left(\frac{(1+\beta)}{2}\right)\times\beta\times2^{(\beta-1)/2}}\right]^{1/\beta} \tag{3}$$

巢穴更新后，用均匀分布的随机数 $R\in(0,1)$ 与蛋被发现概率 Pa 作比较，若 $R\ge Pa$，丢弃不好的巢穴位置，并用式（4）重新构造新的巢穴位置；反之，保留当前位置。位置更新[1]如下：

$$x_{t+1,i}=x_{t,i}+r(x_{t,j}-x_{t,k}) \tag{4}$$

其中：r 表示比例因子，是（0,1）区间的均匀分布随机数；$x_{t,j}$ 和 $x_{t,k}$ 表示 t 代中的两个随机解。

1.2　反向学习

反向学习（opposition-based learning）是 2005 年被 Tizhoosh[14]提出的智能计算领域中的一种新技术。该算法的原理为：基于当前的可行解，同时评估其反向学习的解，并选择更好的可行解[15]。

定义 1 若 $x\in[a,b]$ 之间的任意实数，则 x 的基于反向学习的数如下所示[16]：

$$x'=a+b-x \tag{5}$$

可得，x 到 a 的距离为 $|x-a|=x-a$，根据式（5）可得 x' 到的距离为 $|b-x'|=|b-(a+b-x)|=|a-x|=x-a$，因此，这两个距离是相等的，如图 1 所示。

图 1　任意实数与它的反向学习数

Fig. 1　Real number and its opposition-based learning number

定义 2 若 $p=(x_1,x_2,\ldots,x_D)$ 之间的任意 D 维空间中的点为 $x_i\in[x_{i\min},x_{i\max}]$，则它反向学习的解如下所示[17]：

$$\begin{cases}p'=\left(x_1',x_2',\ldots,x_D'\right)\\x_i'=x_{i\min}+x_{i\max}-x_i\end{cases} \tag{6}$$

可以看出，由于可行解和反向学习的点位于搜索空间的两侧，类似于同时搜索两个对称空间，所以当引入反向学习时，可以扩大搜索空间、提高搜索效率。

2　DA-DOCS 算法

2.1　逐维反向学习策略

在标准 CS 算法中，对于高维复杂函数来说，采用先更新解的所有维度信息，再根据目标函数适应度进行评价，会造成各维度之间干扰，掩盖进化维度的信息，从而影响解的收敛速度与寻优精度。本文提出的逐维反向学习策略，不仅能够有效避免各维度之间相互干扰的现象，而且能扩大种群搜索范围，开采新的搜索空间，增强种群的多样性。在每一代的演化中，它类似于独立每一个维度的进化，并在每个维度同时搜索两个对称区域，提高了算法的收敛效率，也提升了算法的全局寻优能力。

在逐维反向学习策略中，设种群规模为 N，维度为 D，迭代过程选择更新位置时的位置矩阵为 $nest_{n\times Di}$。每个维度的上、下限分别为 $up_{N\times Di}$、$low_{N\times Di}$，则解的每维的基于反向学习的位置矩阵如下：

$$nest_{N\times Di}'=up_{N\times Di}+low_{N\times Di}-nest_{N\times Di} \tag{7}$$

逐维反向学习策略在进行算法的迭代过程中，当被淘汰的蛋被重新构造代替后，与没有被淘汰的蛋汇集，不再使用 CS 算法中的整体更新再评价方式评价，而是对更新后的解的每维进行反向学习，将更新的各维度的值映射到它的反向学习数进行空间搜索，淘汰退化维度信息，不但减少各维度间的干扰，而且增加解空间的搜索范围，从而提高搜索效率和寻优能力。逐维进化学习策略考虑了各维度的更新信息，某一维度的值经过反向学习之后与其他维的值组成新的解，根据目标函数适应度评价该新解，如果能够改善当前解的质量，则保留该维度进行反向学习的更新结果；反之，则放弃对于当前维的更新，保留反向学习之前的维度信息，利用这种精英保留的方式将进行下一维的反向学习更新，直到各维度更新完毕。

基于逐维反向学习策略，强化了进化维度的信息，减少了标准 CS 算法随机更新所消耗的评价次数，提高算法的收敛速度和寻优能力；采用精英保留方式，提高了算法的效

率。

2.2 动态适应

标准 CS 算法迭代过程中，当产生的随机概率 R 大于生存概率 Pa，则蛋被淘汰，将按照式（4）作为依据构造新的蛋来代替被淘汰的蛋。但标准 CS 算法中的构造更新基于解空间的两个随机解，具有较大的随机性和不稳定性，并没有充分利用当前解的信息动态适应迭代更新，不利于逐维反向学习策略进行局部搜索；此外，式（4）中的缩放因子 r 是（0,1）区间的均匀分布随机数，在一定程度上限制了算法的寻优性能。为了提高算法性能，可将式（4）中的缩放因子的取值随迭代次数的动态变化作出适应性调节，r 的取值呈递减趋势，能够在搜索前期跳出局部最优，避免早熟现象，能够在搜索后期增强算法的局部寻优能力。缩放因子充分利用了当前解的信息动态适应，动态引导解的收敛，有利于提高算法的搜索活力。因此，DA-DOCS 算法将式（4）进成如下公式:

$$\begin{cases} x_{t+1,i} = x_{t,i} + r(x_{t,i} - x_{t,k}) \\ r = \dfrac{1}{t_C} \end{cases} \tag{8}$$

其中：$x_{t,k}$ 表示 t 代的随机解；t_c 表示当前迭代次数。

2.3 DA-DOCS 算法流程

基于逐维反向学习的动态适应布谷鸟算法的具体算法步骤主要包括五个部分，其步骤如下：

a) 初始化。

定义目标函数 $F(x)$，设置种群规模为 N、搜索空间维数为 D、最大发现概率 Pa、最大迭代次数为 t_{max} 以及精度阈值 δ 等参数，并随机生成 N 个鸟窝的初始位置 $X_i = (1,2,...,N)$。

b) 随机游走。

选择目标函数并计算每个巢穴位置的适应度 f_x，保留适应度最高的巢穴位置，根据莱维飞行随机游走产生新的巢穴位置。

c) 选择更新。

用均匀分布的随机数 $R \in (0,1)$ 与蛋被发现概率 Pa 作比较，若 $R \geq Pa$，丢弃不好的巢穴位置，用式（8）更新巢穴的位置；反之，保留当前解。计算目标函数的适应度值，将适应度较好的巢穴保留到下一代。

d) 逐维反向更新。

对步骤 c）中保留下来的解，根据式（7）进行逐维反向学习，分别考虑各维度的更新信息。经过反向学习之后的当前维度的值与其他维组成新的解，评价该组合解。若改善当前解的质量，则保留当前维度的更新结果；反之，则放弃当前维的更新，保留未经过反向学习前的当前维度的值，利用精英保留方式进行下一维的反向学习更新，直到各维度更新完毕。

e) 结束判决。

判断当前解的质量是否满足算法结束条件，若满足算法结束条件，则结束，输出最终的解；否则返回步骤 b），直到满足算法结束条件时结束。改进算法流程如图 2 所示。

2.4 改进算法复杂度分析

由文献[18, 19]可知，布谷鸟的搜索算法的时间复杂度为

$$T(n) = 3O(n + f(n)) = O(n + f(n)) \tag{9}$$

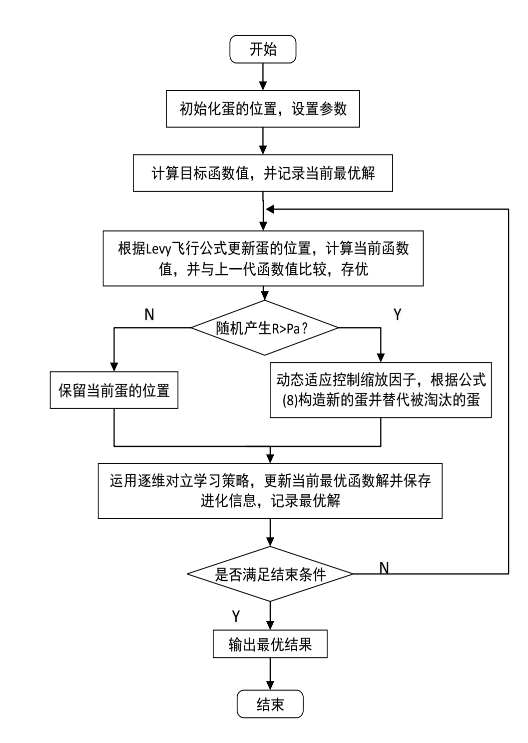


图 2 DA-DOCS 算法流程

Fig. 2 Schematic diagram of DA-DOCS algorithm

布谷鸟搜索算法的时间复杂度取决于计算目标函数的时间复杂度 $f(n)$，当 $f(n)$ 比 n 高阶运算时，复杂度为 $O(f(n))$；当同阶或低阶时，复杂度为 $O(n)$ [20]。根据本文改进算法的流程图 1 可知，引入的逐维反向学习策略，增加了 $O(nf(n))$ 的运算量；同时，为了提高算法的搜索活力，引入的动态适应机制，增加了一层内置迭代，则本文改进算法的时间复杂度为

$$T'(n) = T(n) + O(nf(n)) + O(n) = O(n + nf(n)) \tag{10}$$

另外，空间复杂度 $S(n)$ 主要受种群规模 N 和搜索空间维数 D 的影响，可表示为

$$S(n) = O(f(n)) = O(D \cdot N) \tag{11}$$

可以看出，本文提出的改进算法的算法复杂度较高于标准布谷鸟搜索算法，且维度越高，时间复杂度和空间复杂度越高。

3 实验与结果分析

3.1 测试函数和实验参数设置

本文采用 MATLAB R2016a 进行仿真实验，运行环境为 64 位 Windows 7 操作系统，处理器类型为 Intel Core i5-6500。为了验证 DA-DOCS 算法的性能，实验引入八个标准测试函数验证（表 1）。其中 $F_1 \sim F_3$ 是单峰函数，$F_4 \sim F_8$ 是多峰函数。F_8 在解空间内含有大量局部极小值的多峰函数，不同维度其理论最小值也不相同[21]。

本文采用式（12）适应度误差算法评价算法的精度收敛能力，公式如下[10]：

$$\delta = f(x) - f(x^*) \tag{12}$$

其中：$f(x)$ 表示算法获得的解的适应度；$f(x^*)$ 表示测试函数已知的最优解。式（12）反映了算法的精度收敛能力，函数误差值越小，解的质量越好。

本文主要从两个主要方面对算法进行测试：a）针对改进 DA-DOCS 算法和标准 CS 算法在解的质量、解的收敛速度以及解的维度变化方面进行分析，测试算法的寻优精度和收敛速度以及解决高维度优化问题时的性能；b）针对固定

维度函数进行测试，并与其他改进 CS 算法进行寻优精度和性能的比较。

3.2 与标准 CS 算法的比较

3.2.1 解的质量分析

为了观察并分析 DA-DOCS 算法求解的质量，本文设定种群规模 N=30，最大迭代次数为 5 000 次，被发现概率为 0.25，莱维飞行的步长缩放因子分别为 0.1,1,1.3[22]，测试函数为 $F_1 \sim F_8$。表 2 展示了 DA-DOCS 算法和标准 CS 算法独立运行 30 次实验后的测试结果，分别从平均适应度误差及其标准差两个方面进行对比，最好结果由粗体凸显显示。

由表 2 可以看出，DA-DOCS 算法相比标准 CS 算法总体上改善了解的质量，对单模函数而言，函数解的质量有显著提高；对于有众多局部极值点的多峰函数，DA-DOCS 算法也表现优秀，特别地，CS 算法在 F_3、F_4 函数未能收敛，但 DA-DOCS 算法在 F_4、F_5 函数上获得全局最优解。由此可得，DA-DOCS 算法相比标准 CS 算法具有更好的求解精度。

表 1　测试函数

Table 1　Test functions

函数名	表达式	维度	范围	理论最优值
Spere	$F_1=\sum_{j=1}^{D}(x_j^2)$	20	[-100,100]	0
Quartic	$F_2=\sum_{j=1}^{D} jx_j^4$	20	[-20,20]	0
Rosenbrock	$F_3=\sum_{j=1}^{D-1}\left[100(x_j^2-x_{j+1})^2+(x_j-1)^2\right]$	20	[-100,100]	0
Rastrigin	$F_4=\sum_{j=i}^{D}(x_j^2-10\cos(2\pi x_j)+10)$	20	[-5.12,5.12]	0
Griewank	$F_5=\frac{1}{4000}\sum_{j=i}^{D}x_j^2-\prod_{j=1}^{D}\cos\left(\frac{x_j}{\sqrt{\sqrt{j}}}\right)+1$	20	[-20,20]	0
Ackley	$F_6=-20*\exp\left(-0.2*\sqrt{\sqrt{\frac{1}{n}\sum_{j=1}^{n}x_j^2}}\right)-\exp\left(\frac{1}{n}*\sum_{j=1}^{n}\cos(2\pi x_j)\right)+20+e$	20	[-20,20]	0
Schaffer	$F_7=0.5+\frac{\left(\sin^2\sum_{j=1}^{D}x_j^2-0.5\right)}{\left[1+0.001\left(\sum_{j=1}^{D}x_j^2\right)\right]^2}$	20	[-20,20]	0
Wichalewicz	$F_8=-\sum_{j=1}^{D}\sin(x_j)\left[\sin\left(\frac{jx_j^2}{\pi}\right)\right]^{20}$	10	[0, π]	-9.660151

表 2　CS 与 DA-DOCS 的寻优精度结果

Table 2　Optimization accuracy values of CS and DA-DOCS

函数	维度	CS		DA-DOCS	
		Mean	*Std. Dev*	*Mean*	*Std. Dev*
F_1	20	4.64e-19	1.18e-18	**1.18-103**	**1.51e-103**
F_2	20	4.01e-24	1.20e-23	**1.02e-168**	**7.26e-125**
F_3	20	3.54e+01	2.18e+01	**2.59e-09**	**1.09e-09**
F_4	20	1.24e+02	3.29e+01	**0.00e-00**	**0.00e-00**
F_5	20	2.01 e-03	2.40e-03	**0.00e-00**	**0.00e-00**
F_6	20	8.37e-01	8.87 e-01	**3.55e-15**	**0.00e-00**
F_7	20	9.70e-03	5.10 e-01	**7.97e-04**	**5.76e-04**
F_8	10	2.46e-01	1.54e-01	**2.16e-07**	**4.13e-13**

3.2.2 收敛速度分析

为了观察分析 DA-DOCS 算法的收敛速度，本文设定种群规模 N=30，被发现概率为 0.25，莱维飞行的步长缩放因子分别为 0.1,1,1.3[22]。通过选取的八个测试函数对 DA-DOCS 算法和标准 CS 算法进行 30 次独立实验，分别从收敛于指定精度阈值的平均函数评价次数(FES)及其标准差进行对比（表 3）。其中“—”表示不能收敛到指定阈值，最好结果由粗体凸显显示。图 3 对应八个测试函数寻优收敛曲线图。

标准 CS 算法采用整体评价再更新策略存在维间干扰，在一定程度上浪费了评价次数，未能获得更好的解，使得收敛较慢[10]，但本文改进的逐维反向学习策略在消耗了一定的评价次数之后，增强了局部的求精能力，有利于获得较好的解，从而加快算法的收敛。观察表 3 也可知，通过对相同精度阈值下平均函数评价次数的比较，CS 算法在 F_3、F_4、F_8 测试函数上未能收敛于指定精度阈值；相反，DA-DOCS 算法测试的 $F_1 \sim F_8$ 函数都能够收敛于指定精度阈值，函数平均评价次数及其标准差明显低于标准 CS 算法的函数平均评价次数及其标准差。由此可得，本文改进的算法相比标准 CS 算法有更快的收敛速度和更好的鲁棒性。

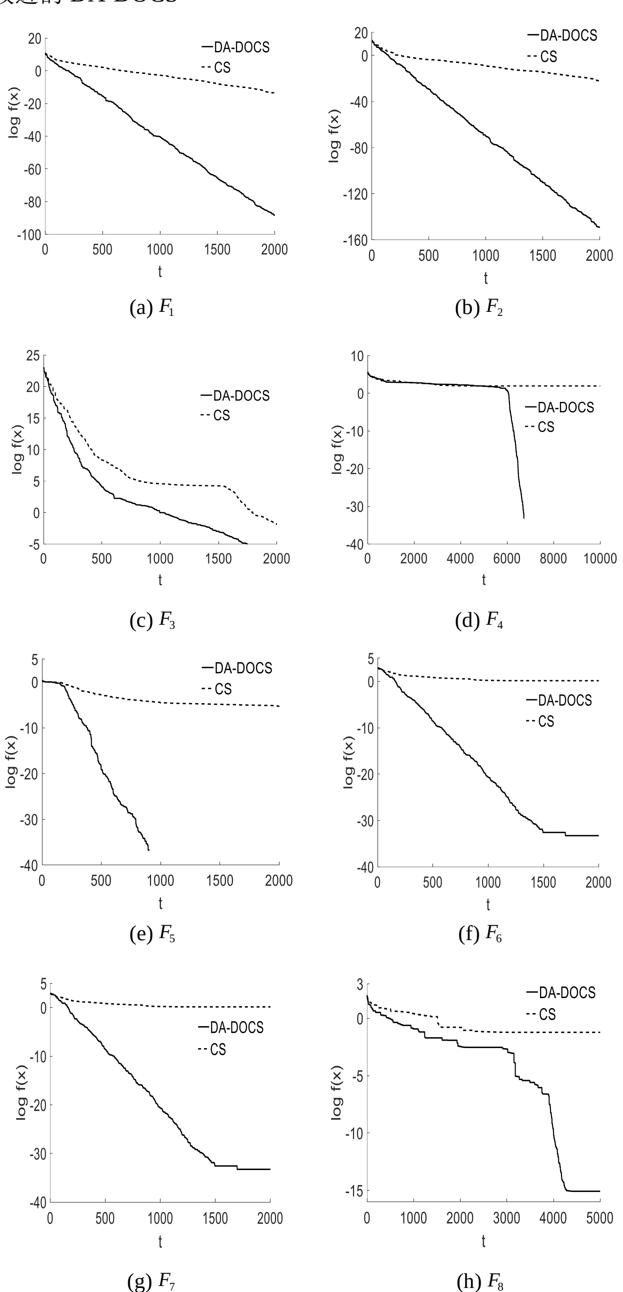
表 3 指定精度阈值下 CS 与 DA-DOCS 的函数评价次数

Table 3　FES of CS and DA-DOCS under specified accuracy threshold

函数	CS		DA-DOCS	
	Mean	*Std. Dev*	*Mean*	*Std. Dev*
F_1	107556	7201	**29044**	**454**
F_2	74364	13868	**18240**	**1065**
F_3	—	—	**86096**	**17356**
F_4	—	—	**30336**	**1714**
F_5	188520	91813	**26884**	**1730**
F_6	244824	67972	**41148**	**609**
F_7	810540	98524	**62136**	**19858**
F_8	—	—	**7240**	**2098**

图 3（a）~（h）以算法的迭代次数为横轴，适应度函数的对数值为竖轴，图形化展示了改进算法和 CS 算法在八个测试函数中的寻优搜索收敛过程，其中，测试函数的维度

为 20。由图 3 可以看出，无论是简单的单峰还是复杂的多峰函数，改进算法相比 CS 算法在收敛前期收敛速度更强，在收敛后期具有更强的搜索活力。特别地，图 3（d）（e）中 DA-DOCS 算法快速收敛到理论最优解。由此可得，本文改进的 DA-DOCS



(a) F_1　(b) F_2　(c) F_3　(d) F_4　(e) F_5　(f) F_6　(g) F_7　(h) F_8

图 3　DA-DOCS 与 CS 寻优收敛曲线图

Fig. 3　Optimization convergence curves of DA-DOCS and CS

算法弥补了标准 CS 算法收敛缓慢，后期局部收敛活力不强的缺点，具有更好的算法挖掘能力和开采能力，验证了表 3 的实验结果。

3.2.3 维度变化分析

为了观察与分析 DA-DOCS 算法随维度变化的影响，本文通过表 4 展示了 CS 算法与 DA-DOCS 算法在不同维度，收敛于同一精度阈值，独立运行 30 次之后获得的平均迭代次数以及标准差。设定种群规模 N=30，被发现概率为 0.25，莱维飞行的步长缩放因子分别为 0.1,1,1.3[22]。其中，“—”表示最大迭代次数 1000×D 内无法收敛到指定精度阈值。最好的结果由粗体凸显显示。

由表 4 可以看出，CS 算法维度从 50 增加到 100 时，函数的平均迭代次数增加 2 倍以上；相较于 DA-DOCS 算法，维度增加时，平均迭代次数增长幅度在 1 倍以内。D=100 维时，CS 算法未能全部收敛，但 DA-DOCS 算法求解的 $F_1\sim F_8$ 函数全部收敛到指定精度阈值。由此可见，DA-DOCS 算法采用逐维反向策略以及动态适应方式，更好地避免了各维度间的干扰，在高维度函数求解中表现出了更好的全局寻优能力，能够提高算法的全局搜索能力和局部开发能力，从而有效提升算法求解的效果、效率以及搜索活力。

表 4　不同维度的迭代次数结果

Table 4　Number of iterations with different dimensions

函数	阈值	CS Mean	CS Std. Dev	DA-DOCS Mean	DA-DOCS Std. Dev
D=50					
F_1	10-5	1985.11	73.01	**794.17**	**19.33**
F_2	10-5	1691.73	75.23	**539.90**	**15.33**
F_3	10-5	18480.00	2332.60	**5012.43**	**1278.60**
F_4	10-4	10373.20	1185.50	**6352.21**	**987.45**
F_5	10-5	1632.2	85.16	**664.00**	**17.92**
F_6	10-5	4835.00	158.25	**1017.90**	**24.37**
F_7	10-3	14663.00	4976.20	**2644.50**	**310.25**
D=100					
F_1	10-5	4049.50	101.48	**1406.11**	**26.04**
F_2	10-5	3945.30	275.62	**1098.30**	**36.39**
F_3	10-5	—	—	**14031.00**	**5761.90**
F_4	10-4	32484.00	66.13	**5812.10**	**1321.45**
F_5	10-5	—	—	**1144.50**	**21.73**
F_6	10-5	—	—	**1732.90**	**38.69**
F_7	10-3	—	—	**6859.10**	**909.68**

表 5　DA-DOCS 与其他改进算法寻优精度结果

Table 5　Optimization accuracy values of DA-DOCS and other improved CS algorithms

函数	参数	DSCS	DDICS	DA-DOCS
F_1	*Mean*	1.15E-61	1.98E-54	**1.01E-103**
	Best	6.97E-62	4.31E-56	**8.62E-104**
	Std. Dev	5.19E-61	1.46E-54	**1.45E-103**
	T(s)	**0.14**	6.18	2.36
F_2	*Mean*	1.13E-117	4.42E-105	**1.16E-168**
	Best	5.97E-118	3.16E-109	**2.65E-168**
	Std. Dev	3.57E-117	5.76E-105	**2.06E-167**
	T(s)	**0.31**	12.99	3.08
F_5	*Mean*	2.51E-03	2.01 E-03	**0.00E-00**
	Best	**0.00E-00**	1.11E-03	**0.00E-00**
	Std. Dev	4.15E-03	2.44 E-03	**0.00E-00**
	T(s)	**0.18**	8.89	3.02
F_6	*Mean*	8.88E-15	5.75E-14	**3.85E-15**
	Best	7.54E-15	**3.21E-15**	3.55E-15
	Std. Dev	2.38E-15	1.77E-15	**5.32E-16**
	T(s)	**0.21**	7.97	2.92
F_7	*Mean*	2.92E-02	2.84E-02	**1.74E-03**
	Best	2.08E-02	4.52E-03	**6.43E-04**
	Std. Dev	3.71E-03	4.70E-03	**5.78E-04**
	T(s)	**0.19**	10.99	2.88

3.3 与其他改进 CS 算法比较

为了进一步证明算法的性能，本文选取了五个测试函数与文献[10]中的 DDICS 算法、文献[22]中的 DSCS 算法进行比较，结果如表 5 所示。表 5 分别从适应度误差平均值、最好值、其标准差以及算法运行时间四个方面进行对比，最好结果用粗体凸显显示。其中种群规模 N=30，被发现概率为 0.25，最大迭代次数为 5 000，维度为 20，每种算法独立运行 30 次，其他改进的 CS 算法参数设置由参考文献确定。图 4 为 DA-DOCS 算法与其他改进 CS 算法的函数寻优收敛曲线图。

由表 5 可以看出，各改进的算法针对不同函数，呈现出来的性能各不一样。DA-DOCS 算法在 F_5 函数上可以收敛到理论最小值，表现出较强的寻优能力，在适应度误差平均值与最大值上，与其他算法相比具有较大优势，针对 F_1 函数，DA-DOCS 算法相比其他改进算法提高了近 40 个等级，由此可得，本文改进的算法相比其他改进的 CS 算法，具有更好的寻优精度。在算法运算时间上，本文改进的算法相比较于 DSCS 算法，所用时间较长，但相比较 DDICS 算法，本文改进的算法所用时间减少了 69.68%，有效缩短了算法运行时间。因此，改进算法表现出了较强的竞争力。由图 4（a）~（e）可以看出，DA-DOCA 算法相对于 DDICS 算法和 DSCS 算法有较好的求解精度以及收敛速度，在收敛后期更具活力，从而验证了表 5 的结果。

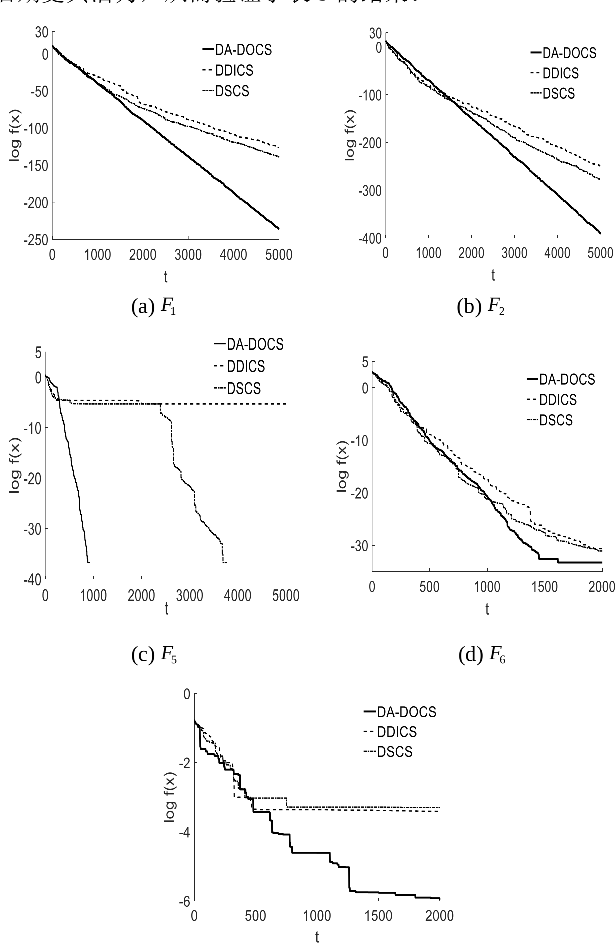


图 4 DA-DOCS 与其他改进 CS 寻优收敛曲线图

Fig. 4 Optimization convergence curves of DA-DOCS and other improved CS

综上所述，对于文中选取的八个测试函数，本文提出的基于逐维反向学习的动态适应布谷鸟算法，采用逐维方向学习和动态适应的方式，减少维间干扰，扩大搜索范围，动态适应控制缩放因子，有效地提高了 CS 算法的收敛速度、求解精度和搜索活力。

4 结束语

本文针对 CS 算法的不足，提出了一种基于逐维反向学习的动态适应布谷鸟搜索算法。该算法主要针对标准 CS 算法做了两点改进：a)在 CS 算法中加入逐维反向学习，并利用精英保留的方式评价该更新结果，增强了种群的多样性，扩大了搜索范围，强化了维度间进化维的信息，减少了维间互扰，使得算法在处理高维度优化函数时，具有更好的寻优精度和搜索速度；b)充分利用当前解的信息进行动态适应的缩放因子控制，使得上一代的最优解的信息能够影响当前解的更新方向，提升解的搜索活力。通过对八个标准测试函数进行实验仿真，结果表明，相比标准 CS 算法，本文改进 DA-DOCS 算法在高维度优化问题时，具有更好的收敛速度和寻优能力以及搜索活力；相比其他改进的 CS 算法，DA-DOCS 算法具有较好的收敛精度和搜索活力，具有一定的竞争力。

参考文献：

[1] Yang Xinshe, Deb S. Cuckoo Search via Levy flights [C]// Proc of World Congress on Nature & Biologically Inspired Computing. Piscataway: IEEE Publications, 2009: 210-214.

[2] Jiang Minlan, Luo Jingyuan, Jiang Dingde, *et al.* A cuckoo search-support vector machine model for predicting dynamic measurement errors of sensors [J]. IEEE Access, 2017, 4 (99): 5030-5037.

[3] Wang Lijing, Zhong Yiwen, Yin Yilong. Nearest neighbor cuckoo search algorithm with probabilistic mutation [J]. Elsevier Science Publishers B. V. Amsterdam, 2016, 49: 498-509.

[4] 兰少峰, 刘升. 布谷鸟搜索算法研究综述 [J]. 计算机工程与设计, 2015, 36 (4): 1063-1067. (Lan Shaofeng, Liu Sheng. Overview of research on Cuckoo search algorithm [J]. Computer Engineering and Design, 2015, 36 (4): 1063-1067.)

[5] 杨辉华, 王克, 李灵巧, 等. 基于自适应布谷鸟搜索算法的 K-means 聚类算法及其应用 [J]. 计算机应用, 2016, 36 (8): 2066-2070. (Yang Huihua, Wang Ke, Li Lingqiao, *et al.* K-means clustering algorithm based on adaptive Cuckoo search and its application [J]. Journal of Computer Applications, 2016, 36 (8): 2066-2070.)

[6] 王凡, 贺兴化, 王燕, 等. 基于 CS 算法的 Markov 模型及收敛性分析 [J]. 计算机工程, 2012, 38 (11): 180-185. (Wang Fan, He Xingshi, Wang Yan, *et al.* Markov model and convergence analysis based on Cuckoo search algorithm [J]. Computer Engineering, 2012, 38 (11): 180-185.)

[7] 陈华, 张艺丹. 基于 logistic 模型的自适应布谷鸟算法 [J]. 计算机工程与应用, 2015, 51 (20): 31-35. (Chen Hua, Zhang Yidan. Adaptive Cuckoo algorithm based on logistic model [J]. Computer Engineering and Applications, 2015, 51 (20): 31-35.)

[8] Pauline O, Zarita Z, Chee K S, *et al.* Adaptive Cuckoo search algorithm with two-parent crossover for solving optimization problems [J]. Advances in Swarm and Computational Intelligence Springer International, 2015, 6 (2): 427-435.

[9] 马卫, 孙正兴, 李俊楼. 基于 Powell 局部搜索策略的全局优化布谷鸟算法 [J]. 计算机应用研究, 2015, 32 (6): 1667-1775. (Ma Wei, Sun Zhengxing, Li Junlou. Cuckoo search algorithm based on powell local search method for global optimization [J]. Application Research of

Computers, 2015, 32 (6): 1667-1775.)
[10] 王李进，尹义龙，钟一文. 逐维改进的布谷鸟搜索算法 [J]. 软件学报, 2013, 24 (11): 2687-2698. (Wang Lijin, Yin Yilong, Zhong Yiwen. Cuckoo search algorithm with dimension by dimension improvement [J]. Journal of Software, 2013, 24 (11): 2687-2698.)
[11] Wang Jun, Zhou Bihua, Zhou Shudao. An improved Cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation [J]. Computational Intelligence and Neuroscience, 2016, 8 (2): 1-8.
[12] 高云龙，闫鹏. 基于多种群粒子群算法和布谷鸟搜索的联合寻优算法 [J]. 控制与决策, 2016, 31 (4): 601-608. (Gao Yunlong, Yan Peng. Unified optimization based on multi-swarm PSO algorithm and Cuckoo search algorithm [J]. Control and Decision, 2016, 31 (4): 601-608.)
[13] Mlakar U, Fister I J, Fister I. Hybrid self-adaptive Cuckoo search for global optimization [J]. Swarm & Evolutionary Computation, 2016, 29: 47-72.
[14] Tizhoosh H R. Opposition-based learning: a new scheme for machine intelligence [C]// Proc of Computational Intelligence for Modelling. Vienna, Austria: Computer Society, 2005, 1: 695-701.
[15] Wei Wenhong, Zhou Jianlong, Fang Chen, *et al*. Constrained differential evolution using generalized opposition-based learning [J]. Acta Electronica Sinica, 2016, 20 (11): 4413-4437.
[16] Zhang Sen, Luo Qifang, Zhou Yongquan. Hybrid grey wolf optimizer using elite opposition-based Learning strategy and simplex method [J]. International Journal of Computational Intelligence and Applications, 2017, 16 (2): 1-12.
[17] Ahandani M A, Alavi-Rad H. Opposition-based learning in shuffled frog leaping: an application for parameter identification [J]. Information Sciences. 2015, 291 (291): 19-42.
[18] 张永韡，汪镭，吴启迪. 动态适应布谷鸟搜索算法 [J]. 控制与决策, 2014, 29 (4): 617-622. (Zhang Yongwei, Wang Lei, Wu Qidi. Dynamic adaptation Cuckoo search algorithm [J]. Control and Decision, 2014, 29 (4): 617-622.)
[19] Elkeran A. A new approach for sheet nesting problem using guided Cuckoo search and pairwise clustering [J]. European Journal of Operational Research, 2013, 231 (3): 757-769.
[20] 肖海林，张文娟，聂在平，等. 基于布谷鸟搜索算法的用户选择和干扰对齐 [J]. 电子科技大学学报. 2017, 46 (6): 801-806. (Xiao Hailin, Zhang Wenjuan, Nie Zaiping, *et al.* User selection based on Cuckoo search algorithm and Interference Alignment [J]. Journal of University of Electronic Science and Technology of China, 2017, 46 (6): 801-806.)
[21] 范帅军. 布谷鸟搜索算法的应用研究与改进 [D]. 成都：西南交通大学, 2016. (Fan Shuaijun. Application research and improvement of Cuckoo search algorithm [D]. Chengdu: Southwest Jiaotong University, 2016.)
[22] Cai Zefan, Yang Xiaodong. Cuckoo Search Algorithm with deep search [C]// Proc of the 3rd IEEE International Conference on Computer and Communications. Chengdu: IEEE Publications, 2018: 2241-2246.